2. Web-ресурс сети Интернет [сайт]. Датчик газа MQ-9. URL: http://autohome.org.ua/gas-sensor-mq9-detail.
3. Web-ресурс сети Интернет [сайт]. Разбираемся с датчиками MQ. URL: http://blog.kvv213.com/2016/09/.

УДК 658.52.56.004.057.5
Г. С. Алексеев, В. В. Лавров, И. А. Гурин
ФГАОУ ВО «Уральский федеральный университет имени первого Президента России Б.Н. Ельцина», г. Екатеринбург, Россия

ИСПОЛЬЗОВАНИЕ МЕТОДОЛОГИИ AGILE В КОМАНДНОЙ РАЗРАБОТКЕ ПРОЕКТА «ЭКОЛОГИЯ» С ПРИМЕНЕНИЕМ СИСТЕМЫ КОНТРОЛЯ ВЕРСИЙ GIT

Аннотация

В работе разработан программный продукт «Экология» для преподавателей и студентов кафедры «Теплофизика и информатика в металлургии». Система предназначена для увеличения эффективности образовательного процесса путем автоматизации рутинных расчетов и визуализации их результатов. Система может быть использована как в образовательных целях, так и в исследовательской деятельности. Однако разработка мощных программных комплексов без четкой концепции и модели затруднена и способна привести к непредвиденным трудностям, поэтому при организации работы коллектива необходимо использовать гибкие методологии разработки, системы контроля версий и управления проектами.
Ключевые слова: Agile, Git, контроль версий.

Abstract

In this paper, we have developed a software product Ecology for teachers and students of the chair of Thermophysics and Informatics in metallurgy. The system is designed to increase the efficiency of the educational process by automating routine calculations and to visualize their results. The system can be used in both educational purposes and research activities. However, the development of large software products without a clear plan of action is difficult, therefore, in the organization of large teams need to use flexible development methodology, version control and project management.
Keywords: Agile, Git, version control.

Разработка программного обеспечения, как правило, сопряжена с несколькими проблемами: эффективность, затратность человеческих ресурсов, надежность. Поэтому правильная организация командной работы остается важнейшим фактором для достижения запланированного результата в установленные сроки. В этом отношении проектная организация предыдущего поколения (т.н. водопадная модель разработки) не является оптимальной, хотя в некоторых процессах эта модель более чем применима. При разработке системы «Экология» было принято решение использовать современную гибкую методологию разработки Agile в купе с системой контроля версий Git.

Для оптимизации работы команды в рамках Agile принято утверждать внутри коллектива определенные роли со своими обязанностями и возможностями. Для работы над проектом «Экология» их было выделено три:

– Team – leader (рис. 1): является руководителем команды внутри проекта, назначает временные рамки, контролирует выполнение задач, закрепленных за каждым разработчиком. Также в обязанности team – leader'а входит сборка проекта «Экология» в единое целое, его

компиляция, разрешение конфликтных ситуаций (ошибки в коде, несогласованность между расширениями файлов и другое);

– разработчик (программист): является «движущей силой» проекта, специалист, занимающийся непосредственным написанием кода и решением поставленных перед ним задач, например, созданием библиотеки классов;

– тестировщик (см. рис. 1, тестировщики выделены зеленым): член команды, отвечающий за работоспособность системы. Занимается выявлением ошибок и проблем в уже написанном коде, посредством модульного тестирования и эталонного расчета, выполненного в электронном редакторе таблиц Excel. Однако исправлением и корректировками занимается непосредственно разработчик или ответственный исполнитель.

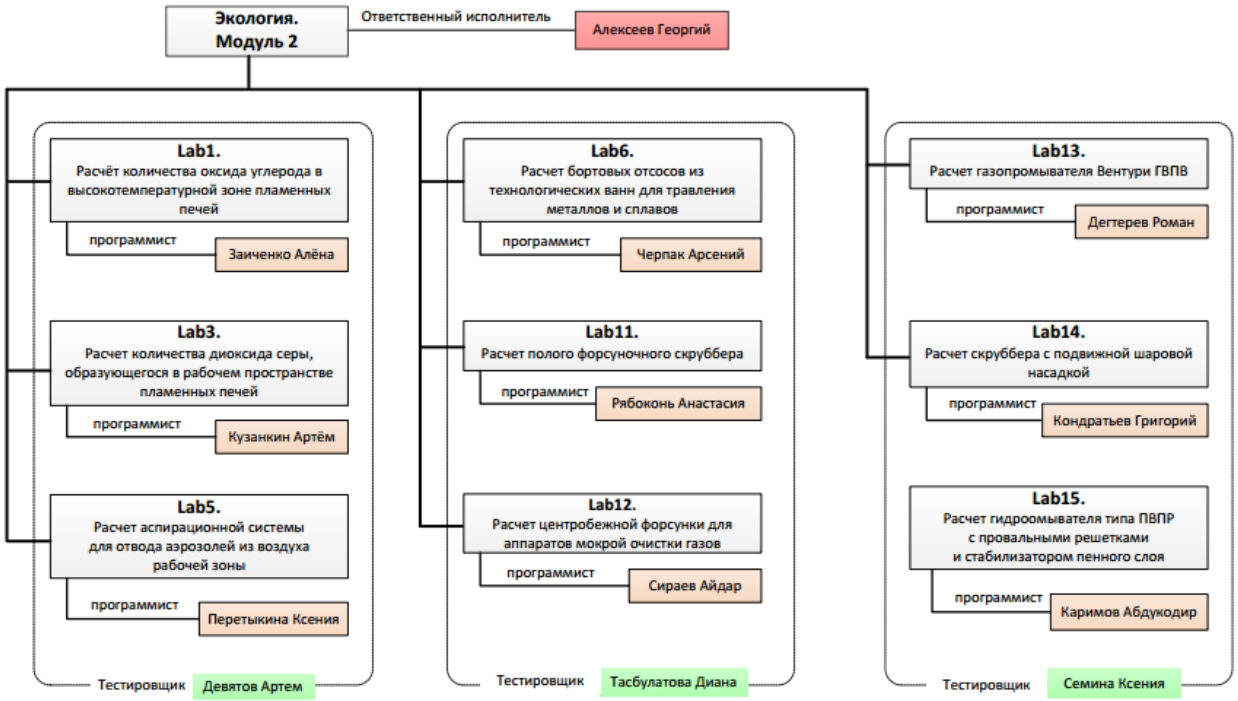

Рис. 1. Распределение ролей при разработке "Модуля 2" проекта "Экология"

Основной временной единицей командной разработки согласно Agile является т.н. спринт – короткий цикл, длящийся, как правило, одну, две недели, по окончании которого заказчику может быть представлена стабильная рабочая и протестированная версия продукта. Функционал на данном этапе может быть еще не до конца завершен. В конце спринта, как правило, проводятся корректировки вектора разработки, смещаются приоритеты в направлении тех или иных задач; во время спринта добавление в план разработки новых идей, редактирование старых не допускается (т.н. принцип «закрытого окна»).

Для достижения гибкого взаимодействия внутри команды необходимы мощные инструменты. Таковыми в рамках проекта «Экология» стали:

– распределенная система контроля версий Git;

– Atlassian Bitbucket – веб-сервис для хостинга проектов и их совместной разработки, основанный на системе контроля версий Git. Именно на этом сервере будет храниться общий репозиторий для работы с проектом (рис. 2);

– Atlassian Jira – платформа управления проектами, task – manager;

– TortoiseGit для удобной работы с Git на локальном компьютере.

Модель разработки выглядит следующим образом (см. рис. 2):

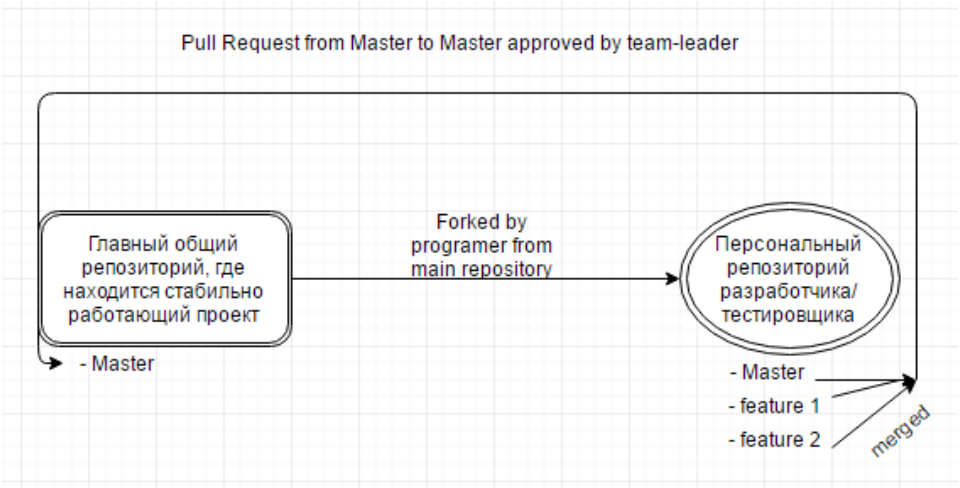

Рис. 2. Схема взаимодействия команды

В первую очередь ответственным исполнителем создается общий репозиторий (от англ. repository – хранилище), который будет наполняться по мере завершения проекта. В нем имеется только одна ветвь разработки по умолчанию – Master, здесь будет находится стабильная протестированная версия проекта без недочетов, ошибок.

Затем разработчик или тестировщик выполняют команду Git fork (с англ. *вилка*), которая скопирует главный репозиторий и поместит его в профиль разработчика/тестировщика для самостоятельной доработки, то есть написания своей части большого проекта; например, таковой может являться лабораторная работа №11 «Расчет полого форсуночного скруббера» или отдельный проект, содержащий в себе модульные тесты для проверки библиотек, созданных программистом.

Нужно также отметить, что внутри своего репозитория разработчик/тестировщик может создавать несколько ветвей для разработки. Как правило, каждая новая ветвь репозитория должна соответствовать указаниям team – leader'а, выраженным в назначенной задаче в системе Jira. После завершения создания функционала, предусмотренного задачей и, соответственно, ветвью, последняя «сливается» в ветку Master при помощи функции Git *merge* (с англ. *сливать*). При каждом внесенном изменении в файлы необходимо создавать т.н. коммиты (от англ *commit* – совершать) – особые отметки, о том, для чего были совершены изменения. Итогом работы программиста становится рабочая версия проекта в ветви Master.

Следующий шаг заключается в том, что разработчику или тестировщику необходимо наполнить общий репозиторий, для чего выполняется команда Git *pull request* (с англ. *отправить запрос*), генерирующая запрос к главному репозиторию на изменение (создание) в нем файлов, запись в который имеет право осуществлять только руководитель проекта. Team – leader просматривает этот запрос, при необходимости находит и устраняет конфликты, после чего принимает *pull request* от разработчика. Система *запрос – утверждение* исключает непредвиденное изменение главного репозитория, что в свою очередь делает маловероятным поломку всей программы в ветви Master главного репозитория.

На заключительном этапе создания проекта руководитель проекта должен создать главную экранную форму, собирающую воедино все модули, созданные программистами и разработчиками (рис. 3). После чего следуют заключительная сборка проекта и релиз.

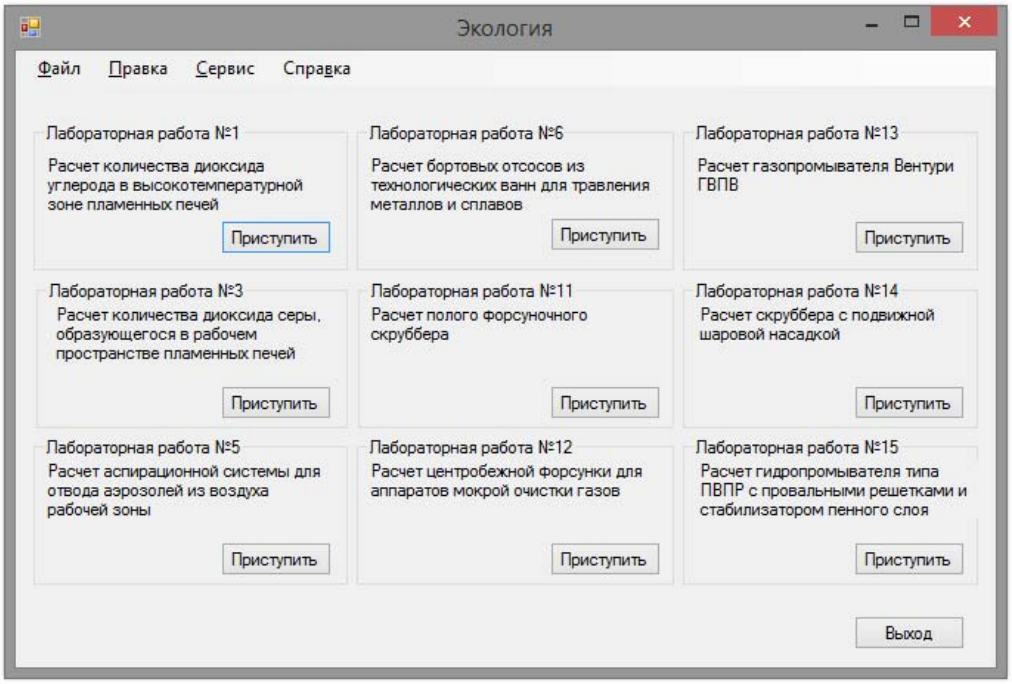

Рис. 3. Эскиз главной экранной формы

Таким образом, используя гибкую методологию разработки Agile, удалось заметно повысить эффективность работы команды, избежать т.н. смертельных маршей, конфликтных ситуаций внутри коллектива.

Список использованных источников

1. Мартин Р.К., Ньюкирк Дж.В., Косс Р.С. Быстрая разработка программ. Принципы, примеры, практика. – М.: Вильямс, 2004. – 752 с.
2. Вольфсон Б. Гибкие методологии разработки. URL: http://adm-lib.ru/books/10/Gibkie-metodologii.pdf.
3. Шакон С. Pro Git. URL: https://git-scm.com/book/ru/v1.

УДК 658.52.56
А. А. Бурыкин, Е. Н. Конюков
ФГАОУ ВО «Уральский федеральный университет имени первого Президента России Б.Н. Ельцина», г. Екатеринбург, Россия

РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ РАСЧЕТА ТРЕХЗОННОЙ МЕТОДИЧЕСКОЙ ПЕЧИ

Аннотация

Статья посвящена разработке программного продукта, который позволит рассчитывать основные показатели работы трехзонной методической печи, предоставлять пользователю результаты расчета в численном и графическом виде.

Ключевые слова: трехзонная методическая печь, расчет показателей, программный продукт.

Abstract

This article was presented development of software for calculating parameters of three-zone methodic furnance. The program shows results in numerical and graphical forms for users.

Keywords: three-zone methodical furnance, parameter's calculating, software.